DOCUMENT RESUME

ED 280 340 HE 020 155

AUTHOR Dukes, Carlton W.
TITLE The Necessity of Training before Education: The Fingertip Effect and Learning as Related to Teaching Computer Skills to Faculty.
PUB DATE Jan 86
NOTE 12p.; Paper presented at the Regional Conference on University Teaching (2nd, Las Cruces, NM, January 8-10, 1986). For complete proceedings, see ED 276 328.
PUB TYPE Viewpoints (120) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *College Faculty; *Computer Assisted Instruction; Computer Literacy; *Computer Science Education; *Faculty Development; Higher Education; *Knowledge Level; Learning Theories; Sequential Approach; *Skill Development

ABSTRACT

Teaching college faculty the principles and applications of computers for computer-based instruction (CBI) in workshops or classrooms is discussed, along with implications of establishing an ongoing program based on sequential levels of knowledge acquisition. Five hierarchical categories for the acquisition of knowledge, based on a model by Hubert Dreyfus, are described: (1) the rule-bound neophyte; (2) the functional operators who have rules memorized by respective functional tasks; (3) the competent systems users who have learned the rules and consult them only for unusual situations; (4) proficient systems developers who are capable of modifying the rules to suit their own purposes; and (5) the expert systems creators. The Project Zero/Harvard Model of learning information processing technology (IPT) is also considered. A computer knowledge acquisition program for faculty is proposed that involves: an organized curriculum to teach according to a sequential model of training; an adequate delivery system, and the immediate availability of a microcomputer. A program outline for teaching IPT to faculty is included. (SW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED280340

THE NECESSITY OF TRAINING BEFORE EDUCATION: THE FINGERTIP EFFECT AND LEARNING AS RELATED TO TEACHING COMPUTER SKILLS TO FACULTY

by

Carlton W. Dukes

New Mexico State University

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Carlton W. Dukes

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

NE 120 155

BEST COPY AVAILABLE

THE NECESSITY OF TRAINING BEFORE EDUCATION:

The Fingertip Effect and Learning

As Related to Teaching Computer Skills To Faculty

I. INTRODUCTION

While fingertip writing, dialing telephones and keying in of figures into calculators have reduced the problems associated with short-term human memory as well as increase the speed of communications, they have also shaped the way in which ideas are formulated. Now, computers have been attributed with certain characteristics which, through the feedback of interactive processing, can imbue the user with opportunities for even more powerful learning and thinking. Some educators expect that learning to use computers and developing even modest programming skills will lead to higher-order intellectual processes. Research at the Harvard Graduate School of Education's Project Zero examined the potential contributions of engaging in programming activities such as analyzing procedural demands, devising a program-like plan and the notion of a "bug"--a host of concepts inviting analogy and application to other procedural contexts (Perkins, 1985).

Word processing too can make substantial structural changes in text both feasible and rapid; data base management systems can potentially reinforce the concepts of classification and relationship, while actual programming is supposed to teach intellectual frameworks of structure, logic and error isolation.

But too often, the exposure to such programmatic content does NOT result in the desired intellectual acquisition. Project Zero suggests that both "high road" and "low road" learning transfer must occur--the mindful abstraction of principles combined with extensive, varied practice before the development of

powerful learning can take place. However, the evaluation of programmatic success tends to be dichotomous in outcome (Adams, 1974) as successful or unsuccessful in terms of taking advantage of available opportunities.

Why opportunities do not get taken may be related to a model by Hubert Dreyfus (1973) of The University of California at Berkeley which describes the level at which the learner is positioned instead of overall success. In this model adaptation, a sequential hierarchy of five levels begins with the neophyte computer user and proceeds through the functional, competent, proficient and expert levels. The differentiating characteristic is the extent to which one is dominated by, uses or creates rules.

By relegating learner success to a binary "yes" or "no" condition, curricula intended to induce higher-order learning through computer usage may fail becaus it may not determine just where on a continuum the learner has actually progressed. In effect, this abandons the student who has made substantial, but as yet incomplete progress on all the tasks or the student who has not reached the "magic cutoff score". Too often the person, not the adequacy of the test of proficiency, is being called into question (Madaus, 1985).

II. Sequential acquisition of knowledge

This paper focuses on experience in teaching college-level faculty the principles and applications of computers for Computer-Based Instruction (CBI) within a workshop and classroom environment, as well as the implications of establishing an ongoing program which is based on sequential levels of knowledge acquisition. It utilizes the five hierarchical categories described by the Dreyfus model which, if used to evaluate the participants position on the hierarchy, could have made the workshops and classroom experiences more effective.

A. The neophyte in a fundamental learning state is characterized by being absolutely rule-bound. Tasks are discrete and discontinuous. In this level, I have observed the participants in a faculty Computer Based Instruction workshop with one finger on the page of an instruction manual, and typing in the designated instructions with another. Even prior to that step, in turning on the computer and monitor, and inserting the diskettes, any interruption or error results in "going back to Square One" and reinitiating the entire "bootup" sequence. I neither expected nor found a comprehension of the actual processes, but a B after A before C series of activities, with expressed high anxiety and error rate.

B. The Functional Operator, has rules memorized by respective functional tasks, knowing the sets of activities/commands needed to start the system, load in a text editor, generate the lesson code and run the lesson presentation module. At this stage, the operator takes for granted the physical activities of turning on and operating the system, and also begins to sense if a sequence is taking too long or a disk drive "doesn't sound right". How things work together becomes more important, with task grouping becoming apparent. The Functional Operator can be characterized by seeking and creating predictability of events. It is usually during crisis cimcumstances such as accidental file or power loss that the operator who can carry out defined functions becomes anxious.

C. The Competent Systems User has by now learned the rules and consults them only for unusual situations, e.g. inserting another file or embedding graphic illustration in the major text material. There is a consistency and regularity in tasks, such as the use of content-free questioning strategies templates created in advance allowing "blank" multiple-choice frames to be copied into the lesson, with all the branching logic and screen formatting already in place. The competent user views the machine and the software processes as a

coherent system, with the emphasis on using them as logical tools to assemble the components of a learning system. At this stage, anxiety often is replaced by curiosity, with few disasters and increasing opportunities for task variability.

D. Proficient Systems Developers are capable of modifying the rules to suit their own purposes, changing the appearance of a screen dramatically through the use of colors or windowing text, animation and graphics. This may involve the use of systems features which are undocumented but are inherent in the software being utilized. Using nested logic resulted in the development of a 16-level menu using logic statements which were either not depicted in the authoring system documentation or which the documentation specifically stated as not being proper.

There is a hierarchy and priority among critical tasks, grouping them together in time or by function. For example, a series of lessons may be created by using a text editor, but not submitted for machine code generation until they are all edited. This not only saves "disk swapping" but allows for similar intellectual activities to occur in clusters.

The Proficient Developer relates various systems to each other, for example using a "screen capture" program to take the report formats from a spreadsheet program, declare it as a file and allow the authoring system software to address it rather than re-type the desired rows and columns as input. The use of a graphic system library permits insertion of attractive detail without having to create it anew for each lesson.

Mere curiosity about "what might happen if..." is replaced by the challenge of developing an intricate, multiple level menu system using simulations, using alternative input devices such as a "joystick" or touch screen.

E. The Expert Systems Creator is a logical outcome of ultimate skill and knowledge. Expert Systems Creators develop their own systems and sets of

rules, deciding the nature of tasks and their sequences, and set up their own system structures. In one case, an expert on the faculty shared only the same physical space as did the others during the workshop, skipping over the materials on text editing and going straight ahead with coding a lesson. There was an occasional question regarding command syntax, and acting entirely independently of the rest of the group, this expert produced a unique adaptation of the graphic commands structure to produce computer lessons involving binary tree traversals programming within an authoring system to produce pseudocode dealing with recursive functions.

This individual had progressed through the first four levels in prior experience bases and compressed days of instruction on authoring systems into less than two hours.

III. Relationship to high/low road mechanisms

It might be helpful at this point to compare the "high road" and "low road" learners to the five acquisition levels. As was mentioned earlier, Project Zero indicates that both "high road" and "low road" learning transfer must occur, mindful abstraction of principles combined with extensive, varied practice before the development of powerful learning can take place. In this investigator's mind, the Neophyte learner and Functional user correspond to "Low Road" transfer, while the Competent user and Proficient developer correspond to "High Road" transfer.

Before one becomes competent and proficient in the development of any computer-based task or system (viewed here as primarily intellectual functions), it is necessary to lay the foundation of skill acquisition garnered through extensive and varied practice. There is no substitute for exposure to many hours of digital representations of intellectual concepts which are presented

through electronic excitation of phosphor.

As for the last stage, this is a category by and of itself--expert creators of systems are relatively rare. There are only a few who create BASIC, FORTRAN, Apple computers, VisiCalcs and inexpensive inference engines. They create the ideas and their applications, they make their own rules and these ideas are so persuasive that others are convinced that they need these concepts.

IV. Training Before Education

The continuing thread throughout all of these stages of skill and knowledge acquisition is in one commonality: the extent to which rules dominate behavior, are utilized or are themselves created.

The equivalent of "low-road" transfer is the neophyte level and is limited totally by rules, while the functional user memorizes them in application clusters. Transfer at this level can be described as training. The equivalent of "high-road" transfer is the competent user who only occasionally consults the rules for unusual situations, as well as the proficient developer of systems who modifies the rules to suit the demands of the task. Transfer at this level can be described as education. The expert creates the rules and is a category beyond the scope of this paper.

If the dichtomous outcome of pass/fail, mastery or non-mastery of "low-road" skills acquisition is used as the measure of success or its lack, this fundamentalist approach will almost always determine that all the neophytes are failures. Even the best functional users will fail at being creative authors of computer-based instruction because they will only be able to follow the rules, however good they may be at doing just that. And unless ever-continuing exposure to development of CBI is facilitated, most faculty can be expected to stay at the first two stages. A major factor to take into consideration is the

level of prior exprience and skills. Since the workshops explicitly were announced as requiring "no computer experience" almost all of the participants had just that level of exposure. Indeed, of the 60 faculty in the workshop series, only ONE out of each of the four CBI workshops has continued on to develop workable CBI programs.

If the continuum criterion is used then the learning and acquisition process is viewed as a hierarchy. Rather than state that the faculty who attended the workshops failed to produce a significant number of CBI lessons, it might be better to recognize that some persons have not progressed far enough to be productive, and that others will not progress beyond the first or second stage. They have not "failed" and while they may never progress beyond those first two stages, they may recognize there are a substantial number of tasks which require so much time to master that pre-existing priorities may never allow adequate time and opportunities to put into CBI format. Unless there is opportunity combined with time availability (and what faculty ever have enough time?) there is likely to be no substantial numbers of CBI programs developed. Even very skilled young programmers, without exception, disclosed enormous time on task. There were no overnight wonders (Perkins, 1985).

Thus, faculty who do not progress into the competent and proficient levels may be successful in that they would, at the Functional User level, still have to invest as much as 400 hours of development time to produce one day of Computer-Based Instruction (Martinetz, 1984) may choose to stay with impact writing on a typewriter. They may also recognize that the return on their investment of time to progress to the competent level where more efficient utilization of information processing skills may be used is minimal, given the time on task that would be inevitably required.

V. A COMPUTER KNOWLEDGE ACQUISITION PROGRAM FOR FACULTY

First, an organized curriculum must be established by which faculty are taught according to a sequential model of training through the neophyte level to that of functional user. A separate course should be implemented for competency leading to proficiency, using the models below.

Second, there must be an adequate delivery system. Faculty cannot be reasonably expected to use their own or departmental computers for student CBI sessions, necessarily meaning obtaining cooperation from the University Computer Center.

Third, there must be a microcomputer immediately available--right on the faculty person's desk. To have to arrange for a session or loan of a PC interrupts idea flow, and worse yet, having to pick up materials and spread them over another functions work area is stultifying to the development of CBI. It may take as much as 100 hours to develop an adequate lesson lasting only one hour from the student's perspective.

VI. Models

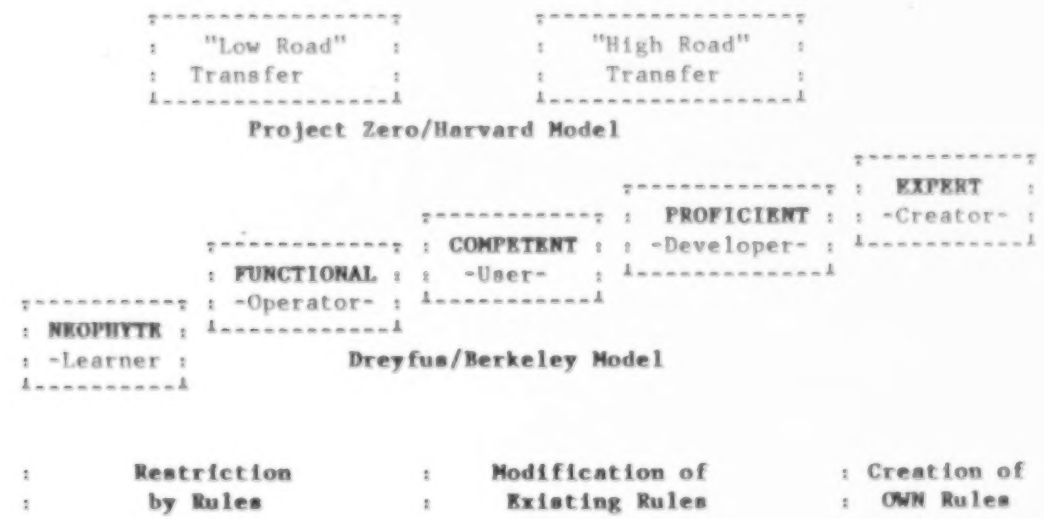

Figure 1. Learning Models for IPT

VII. A Program for Teaching Information Processing Technology to Faculty

Neophyte/Learner and Functional Operator Training

1. Machine fundamentals
   a. CPU/Screen
   b. Disk drive(s)
   c. Keyboard
   d. Printer

2. DOS processes
   a. Format and copy disks
   b. Load/save/copy/rename/ /delete files
   c. Directories/(subs)

3. Word Processing
   a. Insert/delete/move text
   b. Bold/underscore/blink
   c. Center/indent/justify
   d. Heading/footing/paging

4. SpreadSheets
   a. Row/column referencing
   b. Formulas/functions

5. Electronic Filing
   a. Add/change/delete
   b. Index/sort
   c. Retrieve/list/report

6. Programming Computer-Based Instruction
   a. Program editor
   b. Machine lang. generation
   c. Lesson presentation
   d. Student record keeping
   e. Template usage
   f. Graphic components usage

Competent User and Proficient Developer Education

1. Document Management
   a. Outlining/indexing to classify concepts
   b. Screen editing to move text and ideas
   c. Dictionary/thesaurus vocabulary increase
   d. Simultaneous concept retrieval/display of multiple documents

2. Integrated SpreadSheets
   a. Information formats
   b. Formula interactions
   c. Orthogonal versus graphic reports

3. Data Base Management
   a. Classify and cross-classify data
   b. Hierarchies
   c. Relation structuring
   d. Search strategies

4. Programming Computer-Based Instruction
   a. analyze procedural / learning demands
   b. devise a program plan
   c. detect/debug concepts/procedures
   d. formulate questioning tactics
   e. develop teaching strategies

Figure 2. Information Processing Technology Program Outline

REFERENCES

Adams, E. N., 1974. On scoring a mastery learning control test. Journal of Computer-Based Instruction 1(2):50-58.

Dreyfus, Hubert, 1983. Telling what we know: the development of expert systems. Symposium on The Brain and The Mind, Johns Hopkins University. Audio tape NI-831207.02/02-C. 1983 National Public Radio, Washington, D. C.

Jacobson, Robert L., 1985. Microcomputer technology and knowledge of how people think are paving the way for a new era in testing. The Chronicle of Higher Education, XXXI(12):36-37.

Martinetz, Charles, 1984. A step by step approach to course development. Paper presented at a conference of the NSPI, Atlanta, GA.

Perkins, D. N., 1985. The fingertip effect: how information processing technology shapes thinking. Educational Researcher, 14(7), 11-17.